AMENDMENTS TO THE CLAIMS:

This listing of claims will replace all prior versions, and listings, of claims in the application.

Listing of Claims

Claim 1 (Currently Amended): A method of scanning a computer file comprising program code for virus infections, the method comprising:

~~a) identifying program code within the file~~

a) [[b)]] identifying, from a set of predetermined compilers, a [[the]] compiler used to create the program code in the file;

b) [[c)]] determining a [[the]] frequency distribution of selected machine code instructions or sequences of such instructions in the program code in the file; ~~and~~

c) maintaining a database holding frequency distributions of machine code instructions or sequences thereof expected for respective compilers in the set of predetermined compilers; and

d) flagging the file as possibly infected with a virus, or not, on the basis of comparison of the determined frequency distribution with the [[a]] frequency distribution of machine code instructions or sequences thereof stored in the database as being expected for the ~~that~~ compiler identified in step a).

Claim 2 (Currently Amended): A method according to claim 1 wherein step b) [[c)]] comprises a [[the]] step, working from an entry point of the program code, of

b1) tracing an execution graph by decoding successive instruction opcodes and updating frequency counts of decoded instructions as this tracing proceeds.

Claim 3 (Currently Amended): A method according to claim 2 wherein when, during step b1), a subroutine call or conditional branch instruction is encountered, a [[the]] destination of the call or branch instruction is pushed onto a stack, tracing proceeds into the subroutine call, and when a return instruction is encountered, the pushed location is popped from the stack and tracing continues with ~~the~~ following instructions, if any.

Claim 4 (Currently Amended): A method according to claim 1 wherein the program code is examined for opcode constructs, ~~such as subroutine call and subroutine return, instruction sequences~~ which are expected to occur a known ratio to each other and, if the ratio actually found differs from the known one by more than a certain amount, the file is flagged as possibly viral, or subject to further processing.

Claim 5 (Currently Amended): A method according to claim 1, further comprising a ~~and including the~~ step, where step d) flags the file as possibly viral, of

comparing the program code with a list of permissible exceptions and suppressing the flag if the program code is considered to be in the exception list.

Claim 6 (Currently Amended): A computer system operative to scan ~~for scanning~~ a computer file comprising program code for virus infections, the system comprising:

~~a) means for identifying program code within the file~~

~~b) means for identifying~~ a compiler analyser operative to identify, from a set of predetermined compilers, a [[the]] compiler used to create the program code in the file;

~~c) means for determining the~~ an instruction frequency analyser operative to determine a frequency distribution of selected machine code instructions or sequences of such instructions in the program code in the file; ~~and~~

a database holding frequency distributions of machine code instructions or sequences thereof expected for respective compilers in the set of predetermined compilers; and

~~d) means for flagging~~ a frequency distribution checker operative to flag the file as possibly infected with a virus, or not, on the basis of comparison of the determined frequency distribution with the [[a]] frequency distribution of machine code instructions or sequences thereof stored in the database as being expected for the ~~that~~ compiler identified by the compiler analyser.

Claim 7 (Currently Amended): A system according to claim 6, wherein the instruction frequency analyser is operative ~~determining means c) includes tracing means, the tracing means being operable~~, working from an entry point of the program code, to trace an execution graph by decoding successive instruction opcodes and updating frequency counts of decoded instructions as this tracing proceeds.

Claim 8 (Currently Amended): A system according to claim 7, wherein the instruction frequency analyser ~~tracing means~~ is operable such that when a subroutine call or conditional branch instruction is encountered, a [[the]] destination of the call or branch instruction is pushed onto a stack, tracing proceeds into the subroutine call, and when a return instruction is encountered, the pushed location is popped from the stack and tracing continues with [[the]] following instructions, if any.

Claim 9 (Currently Amended): A system according to claim 6, wherein the frequency distribution checker is operative to examine ~~and including means for examining~~ the program code for opcode constructs, ~~such as subroutine call and subroutine return, instruction sequences~~ which are expected to occur a known ratio to each other and, if the ratio actually found differs from the known one by more than a certain amount, the file is flagged as possibly viral, or subject to further processing.

Claim 10 (Currently Amended): A system according to claim 6, further comprising an exception list checker operative, ~~and including means, operable~~ when the frequency distribution checker ~~means d)~~ flags the file as possibly viral, to compare the program code with a list of permissible exceptions and to suppress ~~suppressing~~ the flag if the program code is considered to be in the exception list.

Claim 11 (New): A method according to claim 4, wherein the opcode constructs include subroutine-call and sub-routine return instruction sequences.

Claim 12 (New): A system according to claim 9, wherein the opcode constructs include subroutine-call and subroutine-return instruction sequences.

Claim 13 (New): A computer readable medium having stored thereon instructions for causing a computer to carry out a method for scanning a computer file containing program code for virus infections, the method comprising:

identifying, from a set of predetermined compilers, a compiler used to create the program code in the file;

determining a frequency distribution of selected machine code instructions or sequences of such instructions in the program code in the file;

maintaining a database holding frequency distributions of machine code instructions or sequences thereof expected for respective compilers in the set of predetermined compilers; and

flagging the file as possibly infected with a virus, or not, on the basis of comparison of the determined frequency distribution with the frequency distribution of machine code instructions or sequences thereof stored in the database as being expected for the identified compiler.

Claim 14 (New): A system comprising a computer-readable medium according to claim 13.